Distributed Problem Solving Environment, collaborative workbenches/frameworks and Portals:

how they will be used in the future?

Dennis Gannon
Department of Computer Science
Indiana University

Outline

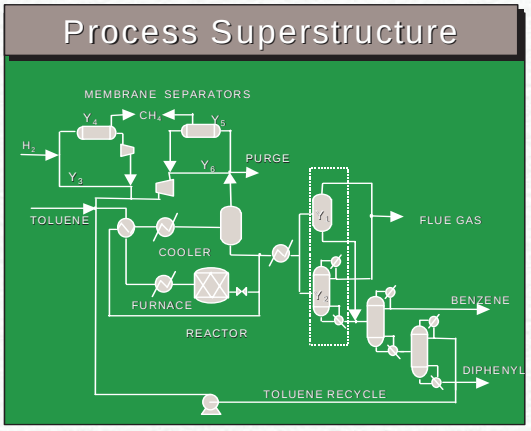
- Collaborative, Distributed Application
  - Chemical Engineering
  - X-Ray Crystallography
- The Technologies
  - Science Portal Project:
    - Java Servlets, Beans
    - Java COG Kit
  - Common Component Architecture
    - A DOE (mostly) effort to build a component frame work for distributed science
  - Other common technologies
    - CORBA, Jini
  - Peer-to-Peer and Grid computing

Overview

- Application and goals
  - X-ray crystallography; remote instrument control and collaboration
- Software architecture
  - Explore use of DoE Common Component Architecture framework
  - Extensions to non-compute components, network awareness
  - Explore use of CoG Kits (commodity Grid components)
- Network requirements and research
  - End-to-end reservation

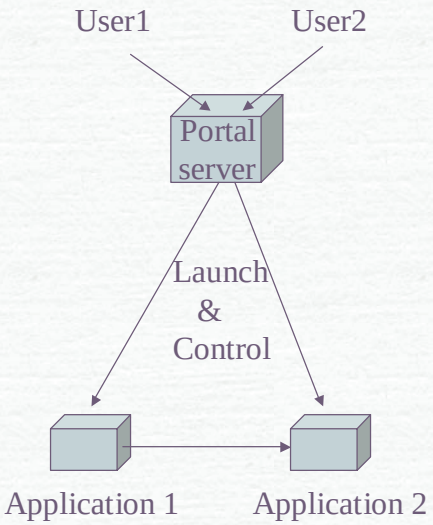
Chemical Engineering & Semiconductor Manufacturing

- Rapid Design
  - Building a chem factory by "drop & drag" software component composition.
  - Each component a proxy for a simulation or database or experimental resource
- Collaboration
  - A web-based workbench.
  - secure access to encapsulated proprietary codes and collaborations

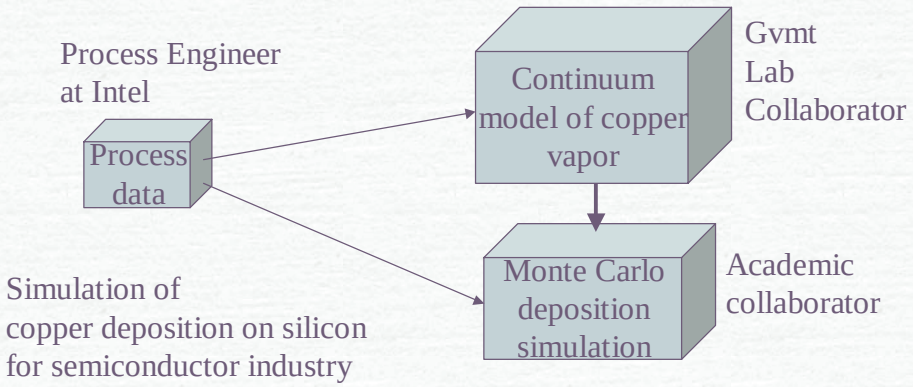


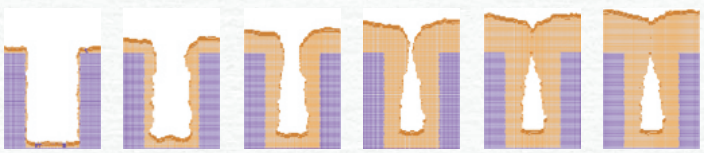
Requirements: Science Portal

- Portal
  - means to collaborate
  - simple interface to complex problem
- Access to Distributed Applications
  - individual components provided by each participant
  - some provide algorithms
  - some data/instruments



Coupled Application





Security Issues

- Raised by Users
- Trust
  - how can the Intel Engineer protect proprietary process data?
  - How can Government Lab employee protect proprietary algorithms?
  - Can my program execute in your environment without leaving "an impression"?
- Encapsulation helps
  - Expose only the interfaces
- Ironclad VM? Can we trust your OS?

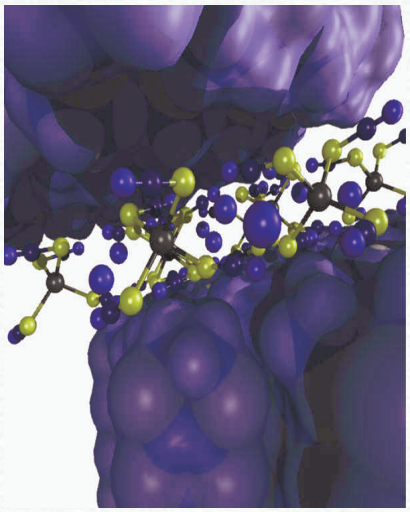
X-Ray Crystallography

Xports:
Network-Based
Macro-Molecular
Structure Determination

R. Bramley, D. F. McMullen,
J. Huffman
Indiana University
I. Foster, G. von Laszewski,
M. Westbrook
Argonne National Laboratory
E. Westbrook,
Lawrence Berkeley Laboratory

Application and Goals

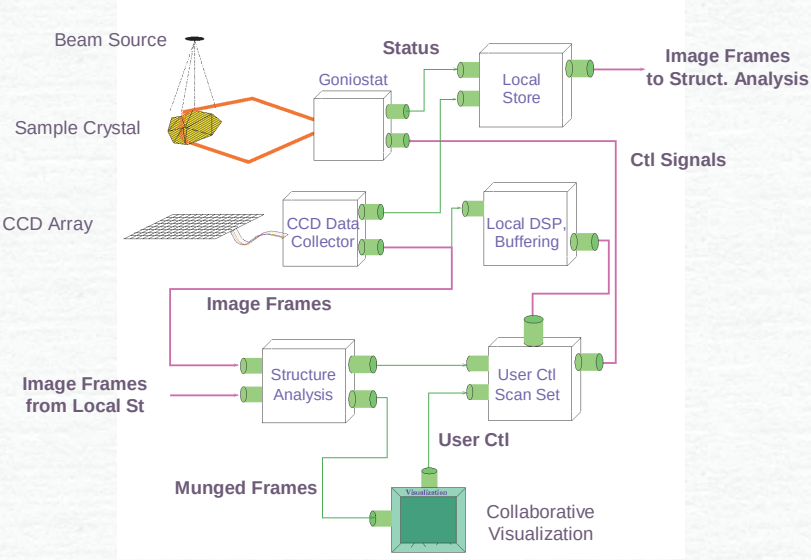
- X-Ray crystallography to determine molecular structure
- Better resolution from high-energy, high-flux beam lines at LBL and ANL
- Current procedure:
  - Ship crystal sample to LBL
  - Lab tech mounts sample, runs predetermined scan procedure
  - Data collected, written to DAT, shipped FedEx
  - If sample is flawed/mounted incorrectly, multi-hour run wasted



Application and Goals

- Desired procedure
  - Scientist sends sample to LBL
  - With scientist's advice via teleconferencing, technician mounts sample
  - Scientist sends initial set of scan commands
  - Preliminary frames are filtered via local/remote components and sent to scientist who manipulates images to check
    - mounting is proper
    - sample is still good
    - whether sample is subject to twinning
  - Scientist sends new scan parameters - or terminates beam-line session
  - At any time, scientist can allow remote colleagues to view partial results and confer

Application



Xport Requirements

- Multiple streams:
  - CCD frame transfer: initial frames of critical importance
  - 2 to 3 video cameras: operator, crystal, crystal alignment, user(s) and Audio
  - Real-time visualization and shared virtual spaces (caves)
- Co-allocation of multiple service levels
  - Video and audio
  - Bulk data transfer to/from HPSS and local cache
  - Data transfer between components during execution of distributed computations
- Security Issues
  - Safety - remote control of complex instruments
    - who has it and what harm can they do.
  - Information leaks -
    - what is cost of encripting data on wire?
    - How is it protected in mass storage systems?

Component Architectures

- What is a CA?
  - A Software Engineering Methodology/Standard
    - to promote code reusability & reduce application complexity.
    - It is a precise protocol to define interoperability between code modules.
  - A Component is-
    - An encapsulated software module defined by its public interfaces.
    - It follows a strict set of behavior rules defined by the architecture.
  - A Component framework is -
    - A compile-time/runtime environment for instantiating, composing and running components.

Examples of CAs:

- Desktop Software:
  - Simplify the design and interoperability of desktop apps
    - Microsoft COM, Java Beans, Gnome CORBA Components
- Distributed Software:
  - Make it possible to easily incorporate a remote object into a local computation
    - Enterprise Java Beans, DCOM, CORBA Component Model (CCM).

A component can be ...

- A desktop utility
  - visualization tool, spreadsheet, matlab, python, ...
- A scalable parallel application written in MPI or other parallel language
  - communication between such components should be parallel.
- The interface to an instrument
  - sensor, telescope, wind tunnel, satellite, etc.
- A database or data archive.
- A linear solver, preconditioner, analysis tools
  - for example, a netsolve interface or proxy.

The DOE Common Component Architecture Project

- A CA for large scale scientific computation
  - Component Characteristics
    - May be SPMD or multi-threaded parallel objects.
  - Heterogeneity
    - Parallel platforms to desktops and any language.
  - Local and Remote
    - Parallel communication for remote parallel interfaces and 0-copy in-process connection.
  - Dynamic Composition and Integration
    - hot-swapable components, shared instances

Building Applications by Composition

- Connect uses Ports to Provides Ports.

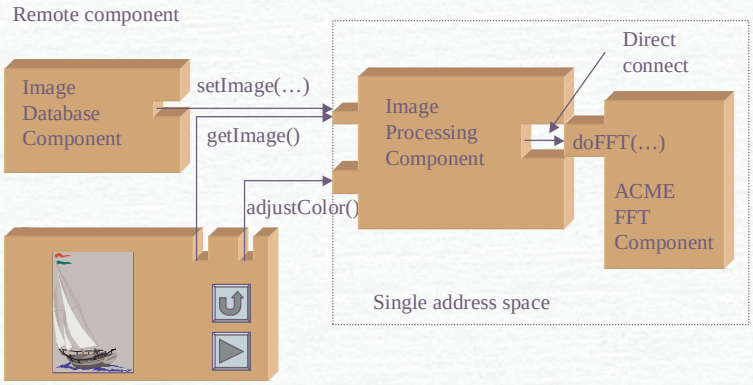
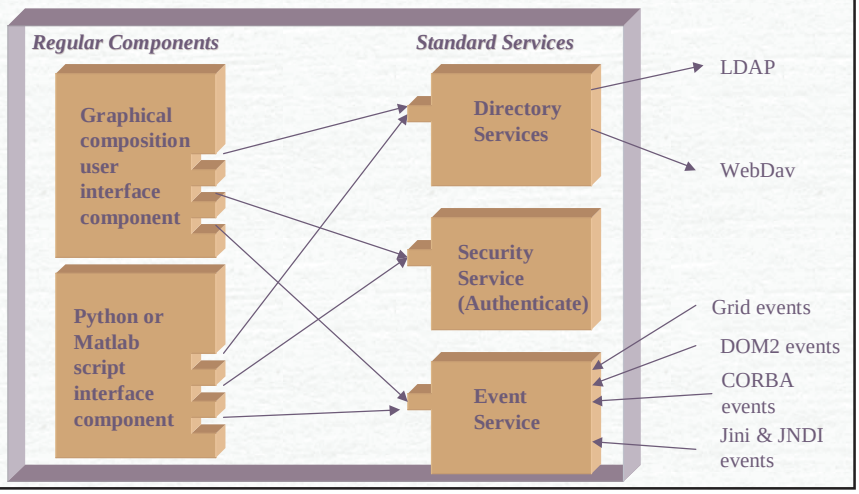


Image tool graphical interface component

A View of the Framework

- Standard Services appear as components



Relationship to Other Standards

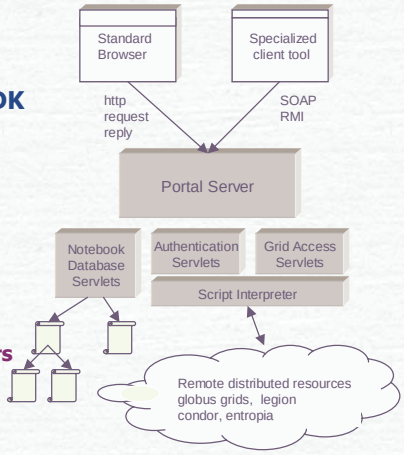
- CORBA
  - CORBA provides a rich service model and distributed object model, specialized vertical market support
  - no co-scheduling, weak information services
- Jini
  - discovery service, events, transactions, leasing
  - all java - local area design
- Enterprise Java Beans & Corba Components
  - good for building some distributed applications
  - missing many of the core services of the grid.

Science Portals

- Objective
  - Setting up distributed applications is hard.
  - To provide a web-based environment for users of a class of related applications to
    - Execute the apps by filling in web form information such as
      - application parameter values
      - path to input files
    - ASP model: user doesn't care where it runs. Just wants it run fast.
  - The ability to compose apps or to script parameter studies
  - A Repository for managing experimental sessions.
  - Both collaborative version and MyServer version

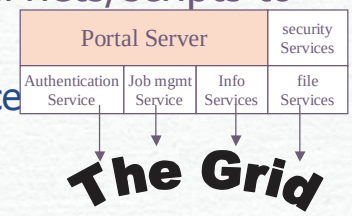
NCSA Alliance Portal Server

- An extension of User Grid Portal
- A Script Engine
  - Python based script pages
  - Scripts can access COG/GPDK to launch & compose jobs, CCA components, etc.
- A Database of user experiment metadata
  - Each session/experiment is saved as a directory containing
    - Scripts used and parameters
    - output pages
    - user annotations
    - Event log
- Simple Grid Event model based on SOAP.



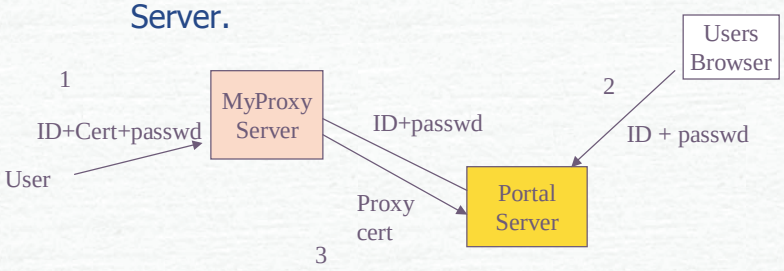
Portal Services

- The portal has web pages/servlets/scripts to
  - Launch Jobs using globus
  - Consult grid information service
  - Manage Remote Files
- Core Technologies:
  - Argonne COG Kit
  - ANL/LBL/NCSA MyProxy Authentication
  - LBL Grid Portal Beans
  - NPACI Hot Page Scripts



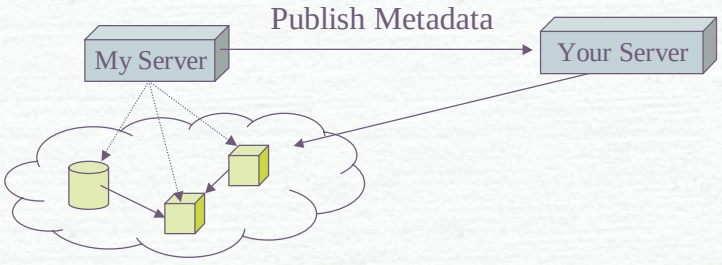
MyProxy Portal Access/Authentication Model

- Three Steps for Authentication
  - 1. Log into a globus client node and create a proxy cert that is stored in the MyProxy cert server with a one-time password.
  - 2. Connect from a web browser to Portal Server and log in with temporary passwd.
  - 3. Portal Server fetches your proxy from MyProxy Server.



Peer-to-Peer Model

- The portal is a evolving to be a "personal server"
  - Each user can run a copy of the server to publish and share access to distributed computations



Conclusions & Observations

- Scientific Apps are not designed with security in mind.
  - Yet it becomes a big issue in collaborations
- Distributed Component Model provides a place to start
  - Encapsulation. Expose only those interfaces you wish to provide as services.
  - Must incorporate client authentication into framework.
- Enterprise Frameworks (EJB) focus security on transactional interactions

Conclusions & Observations

- Portal Technology leverages both Grid and Commodity portal technology
  - GSI/Globus security
  - MyProxy
  - HTTPS
- Dangers lurk here.
  - Future apps will be distributed and collaborative.
  - They will rely on Grid services for access to information and computation.
  - Peer-to-Peer model emerging.

Application Scripting

- Some Apps require linking together several sub-apps.
  - Chem-Eng: monte carlo + Finite Diff codes
- Requires two levels of scripts
  - configuration and launch scripts running in server
  - application proxy scripts
    - monitor sub-app
    - send events
    - stage files